In the Claims:

Please amend claims 1, 11 and 52, and please cancel claims 9, 10, 59, 113 and 118, as indicated below.

1. (Currently amended) A method for providing a linearizable multi-compare, single-swap facility for concurrent software, the method comprising:

snapshotting a plurality of application values corresponding to a respective plurality of targeted memory locations to determine whether any of the application values are changed between two successive reads at the targeted memory locations, wherein said snapshotting comprises reading each of the plurality the application values at least twice and comparing each application value across pairs of successive reads;

updating a first application value corresponding to a first targeted memory location only if said snapshotting indicates that the plurality of application values remain unchanged between two successive reads at the targeted locations, wherein the first targeted memory location is distinct from the plurality of targeted locations; ~~and~~

using a pair of single-location synchronizations to ensure that the first application value remains unchanged across said snapshotting; and

displacing the first application value from the first targeted location prior to a linearization point of the snapshotting, wherein the displacing includes:

reading the first application value;

storing the read value in an auxiliary location associated with a tagged id; and

storing, using a first of the single-location synchronizations, the tagged id in the first targeted location.

2. (Previously presented) The method of claim 1,

wherein the first targeted location and at least one of the plurality of targeted locations are non-contiguous.

3. (Previously presented) The method of claim 1,

wherein a first one of the single location synchronizations precedes the snapshotting; and

wherein a second one of the single location synchronizations follows the snapshotting.

4. (Previously presented) The method of claim 3,

wherein the second one of the single target synchronizations effectuates the updating.

5. (Original) The method of claim 1,

wherein the single-location synchronizations retry on failure.

6. (Original) The method of claim 1,

wherein the method has a non-blocking property.

7. (Previously presented) The method of claim 6,

wherein the non-blocking property includes obstruction-freedom.

8. (Original) The method of claim 1,

wherein the single-location synchronizations employ tagged id displacement for ABA avoidance.

9. – 10. (Canceled)

11. (Currently amended) The method of claim 1 [[9]], wherein the displacing is performed by a load-linked sequence that employs one of the single-location synchronizations.

12. (Previously presented) The method of claim 1, wherein the snapshotting includes:

collecting (i) application values associated with each of the plurality of targeted locations and (ii) tagged ids, if any, from corresponding tagged id locations until two successive collections indicate identical respective application values and identical respective tagged ids.

13. (Previously presented) The method of claim 1, further comprising:

resetting any particular targeted location, including the first targeted location and any of the plurality of targeted locations, in connection with retrieval of an associated application value from a corresponding auxiliary location,

the resetting including displacing, using a single-location synchronization, a tagged id stored in the particular targeted location with the associated application value.

14. (Original) The method of claim 1,

wherein any particular application value is read either from a corresponding one of the targeted locations, if encoded therein, or from an auxiliary location associated with an id, if instead, the id is encoded therein.

15. (Previously presented) The method of claim 1,

wherein the first targeted location and the plurality of targeted locations are non-contiguous.

16. (Original) The method of claim 1,

wherein the single-location synchronizations are compare-and-swap (CAS) synchronizations.

17. (Original) The method of claim 1,
wherein the single-location synchronizations are atomic read-modify-write synchronizations.

18. (Original) The method of claim 1,
wherein the single-location synchronizations employ respective pairs of load-linked (LL) and store-conditional (SC) operations.

19. (Previously presented) The method of claim 1,
wherein the single-location synchronizations are compare-and-swap (CAS) synchronizations employed to define a load-linked (LL) sequence and a store-conditional (SC) sequence, respectively.

20. (Previously presented) The method of claim 19,
wherein the load-linked and store-conditional sequences employ tagged id displacement for ABA avoidance.

21. (Previously presented) A system, comprising:

a processor; and

memory coupled to the processor, wherein the memory comprises program instructions executable by the processor to implement:

a k-compare, single-swap synchronization, wherein to implement the k-compare, single-swap synchronization the program instructions are

configured to perform at least one and no more than two (2) atomic, single-location read-modify-write synchronizations;

an update mechanism configured to update a state of a first targeted location; and

a snapshot mechanism configured to:

verify a state of k-1 other targeted locations, wherein k is greater than 1; and

collect (i) application values associated with each of the other targeted locations and (ii) tagged ids, if any, from corresponding tagged id locations until two successive collections indicate identical respective application values and identical respective tagged ids.

22. (Previously presented) The system of claim 21,

wherein the single-location read-modify-write synchronizations are configured to employ tagged id displacement for ABA avoidance.

23. (Canceled)

24. (Previously presented) The system of claim 21,

wherein the program instructions are further configured to displace, prior to linearization point of the snapshot mechanism, a first application value from the first targeted location.

25. (Previously presented) The system of claim 24, wherein the displacing includes:

reading the first application value;

storing the read value in an auxiliary location associated with a tagged id; and

storing, using a first of the single-location read-modify-write synchronizations, the tagged id in the first targeted location.

26. (Previously presented) The system of claim 24,

wherein the displacing is performed by a load-linked sequence that employs one of the single-location read-modify-write synchronizations.

27. (Canceled)

28. (Previously presented) The system of claim 21,

wherein the implementation resets any particular targeted location, including the first targeted location and any of the k-1 other targeted locations, in connection with retrieval of an associated application value from a corresponding auxiliary location; and

wherein the resetting includes displacing, using a single-location read-modify-write synchronization, a tagged id stored in the particular targeted location with the associated application value.

29. (Previously presented) The system of claim 21,

wherein the program instructions are further configured to read any particular application value either from a corresponding one of the targeted locations, if encoded therein, or from an auxiliary location associated with an id, if instead, the id is encoded therein.

30. (Previously presented) The system of claim 21,

wherein the first and other targeted locations are non-contiguous.

31. (Previously presented) The system of claim 21,

wherein the single-location ready-modify-write synchronizations are compare-and-swap (CAS) synchronizations.

32. (Previously presented) system of claim 21,

wherein the single-location read-modify-write synchronizations comprise respective pairs of load-linked (LL) and store-conditional (SC) operations.

33. (Previously presented) The system of claim 21,

wherein the single-location read-modify-write synchronizations are compare-and-swap (CAS) synchronizations employed to define a load-linked (LL) sequence and a store-conditional (SC) sequence, respectively.

34 – 51. (Canceled)

52. (Currently amended) A computer program product embodied in one or more computer readable media and encoding at least a portion of a synchronization operation, the product comprising program instructions configured to implement:

a snapshot sequence, comprising snapshotting a plurality of application values corresponding to a respective plurality of targeted memory locations to determine whether any of the application values are changed between two successive reads at the targeted memory locations, wherein said snapshotting comprises reading each of the plurality the application values at least twice and comparing each application values across pairs of successive reads;

a linearizable multi-compare, single swap synchronization, comprising updating a first application value corresponding to a first targeted location only if said snapshotting indicates that the plurality of application values remain unchanged between two successive reads at the targeted locations, wherein the first targeted memory location is distinct from the plurality of targeted locations; and

wherein the multi-compare, single swap synchronization further comprises employing a pair of single-location synchronizations to ensure that the application value corresponding to the first targeted location remained unchanged at a linearization point of the snapshot;

wherein the multi-compare, single swap synchronization resets any particular targeted location, including the first targeted location and any of the other targeted locations, in connection with retrieval of an associated application value from a corresponding auxiliary location,

the resetting including displacing, using a single-location synchronization, a tagged id stored in the particular targeted location with the associated application value.

53. (Previously presented) The computer program product of claim 52,

wherein, in an uncontended execution thereof, the multi-compare, single swap synchronization employs no additional single-location synchronizations other than the pair of single-location synchronizations.

54. (Original) The computer program product of claim 52,

wherein the single-location synchronizations employ tagged id displacement for ABA avoidance.

55. (Previously presented) The computer program product of claim 52,

wherein the program instructions are further configured to implement displacing, prior to a linearization point of the snapshot sequence, the first application value from the first targeted location.

56. (Original) The computer program product of claim 55, wherein the displacing includes:

reading the first application value;

storing the read value in an auxiliary location associated with a tagged id; and

storing, using a first of the single-location synchronizations, the tagged id in the first targeted location.

57. (Original) The computer program product of claim 55,
wherein the displacing is performed by a load-linked sequence that employs one of the single-locations synchronizations

58. (Original) The computer program product of claim 52,
wherein the snapshot sequence collects (i) application values associated with each of the other targeted locations and (ii) tagged ids, if any, from corresponding tagged id locations until two successive collections indicate identical respective application values and identical respective tagged ids.

59. (Canceled)

60. (Previously presented) The computer program product of claim 52,
wherein the program instructions are further configured to implement reading any particular application value either from a corresponding one of the targeted locations, if encoded therein, or from an auxiliary location associated with an id, if instead, the id is encoded there.

61. (Original) The computer program product of claim 52,
wherein the first and other targeted locations are non-contiguous.

62. (Original) The computer program product of claim 52,
wherein the single-location synchronizations are read-modify-write synchronizations.

63. (Previously presented) The computer program product of claim 52,
wherein the single-location synchronizations are compare-and-swap (CAS) synchronizations.

64. (Original) The computer program product of claim 52,

wherein the single-location synchronizations employ respective pairs of load-linked (LL) and store-conditional (SC) operations.

65. (Original) The computer program product of claim 52,

wherein the single-location synchronizations are compare-and-swap (CAS) synchronizations employed to define a load-linked (LL) sequence and a store-conditional (SC) sequence, respectively.

66. (Previously presented) The computer program product of claim 52,

wherein the computer readable media include at least one medium selected from the set of a disk, tape or other magnetic, optical, or electronic storage medium.

67. – 110. (Canceled)

111. (Previously presented) A system, comprising:

a processor; and

memory coupled to the processor, wherein the memory comprises program instructions executable by the processor to implement:

a k-compare, single-swap synchronization, wherein to implement the k-compare, single-swap synchronization the program instructions are configured to perform at least one and no more than two (2) atomic, single-location read-modify-write synchronizations;

an update mechanism configured to update a state of a first targeted location; and

a snapshot mechanism configured to verify a state of k-1 other targeted locations, wherein k is greater than 1;

wherein the program instructions are further configured to displace, prior to linearization point of the snapshot mechanism, a first application value from the first targeted location, wherein the displacing includes:

reading the first application value;

storing the read value in an auxiliary location associated with a tagged id; and

storing, using a first of the single-location read-modify-write synchronizations, the tagged id in the first targeted location.

112. (Previously presented) A system, comprising:

a processor; and

memory coupled to the processor, wherein the memory comprises program instructions executable by the processor to implement:

a k-compare, single-swap synchronization, wherein to implement the k-compare, single-swap synchronization the program instructions are configured to perform at least one and no more than two (2) atomic, single-location read-modify-write synchronizations;

an update mechanism configured to update a state of a first targeted location; and

a snapshot mechanism configured to verify a state of k-1 other targeted locations, wherein k is greater than 1;

wherein the program instructions are further configured to reset any particular targeted location, including the first targeted location and any of the k-1 other targeted locations, in connection with retrieval of an associated application value from a corresponding auxiliary location; and

wherein the resetting includes displacing, using a single-location read-modify-write synchronization, a tagged id stored in the particular targeted location with the associated application value.

113. (Canceled)

114. (Previously presented) A method for providing a linearizable multi-compare, single-swap facility for concurrent software, the method comprising:

snapshotting a plurality of application values corresponding to a respective plurality of targeted memory locations to determine whether any of the application values are changed between two successive reads at the targeted memory locations, wherein said snapshotting comprises reading each of the plurality the application values at least twice and comparing each application value across pairs of successive reads;

wherein the snapshotting includes collecting (i) application values associated with each of the plurality of targeted locations and (ii) tagged ids, if any, from corresponding tagged id locations until two successive collections indicate identical respective application values and identical respective tagged ids;

updating a first application value corresponding to a first targeted memory location only if said snapshotting indicates that the plurality of application values remain unchanged between two successive reads at the targeted locations, wherein the first targeted memory location is distinct from the plurality of targeted locations; and

using a pair of single-location synchronizations to ensure that the first application value remains unchanged across said snapshotting.

115. (Previously presented) A method for providing a linearizable multi-compare, single-swap facility for concurrent software, the method comprising:

snapshotting a plurality of application values corresponding to a respective plurality of targeted memory locations to determine whether any of the application values are changed between two successive reads at the targeted memory locations, wherein said snapshotting comprises reading each of the plurality the application values at least twice and comparing each application value across pairs of successive reads;

updating a first application value corresponding to a first targeted memory location only if said snapshotting indicates that the plurality of application values remain unchanged between two successive reads at the targeted locations, wherein the first targeted memory location is distinct from the plurality of targeted locations;

using a pair of single-location synchronizations to ensure that the first application value remains unchanged across said snapshotting;

resetting any particular targeted location, including the first targeted location and any of the plurality of targeted locations, in connection with retrieval of an

associated application value from a corresponding auxiliary location, the resetting including displacing, using a single-location synchronization, a tagged id stored in the particular targeted location with the associated application value.

116. (Previously presented) A computer program product embodied in one or more computer readable media and encoding at least a portion of a synchronization operation, the product comprising program instructions configured to implement:

a snapshot sequence, comprising snapshotting a plurality of application values corresponding to a respective plurality of targeted memory locations to determine whether any of the application values are changed between two successive reads at the targeted memory locations, wherein said snapshotting comprises reading each of the plurality the application values at least twice and comparing each application values across pairs of successive reads;

a linearizable multi-compare, single swap synchronization, comprising updating a first application value corresponding to a first targeted location only if said snapshotting indicates that the plurality of application values remain unchanged between two successive reads at the targeted locations, wherein the first targeted memory location is distinct from the plurality of targeted locations;

wherein the multi-compare, single swap synchronization further comprises employing a pair of single-location synchronizations to ensure that the application value corresponding to the first targeted location remained unchanged at a linearization point of the snapshot;

wherein the program instructions are further configured to implement displacing, prior to a linearization point of the snapshot sequence, the first application value from the first targeted location, wherein the displacing includes:

reading the first application value;

storing the read value in an auxiliary location associated with a tagged id; and

storing, using a first of the single-location synchronizations, the tagged id in the first targeted location.

117. (Previously presented) A computer program product embodied in one or more computer readable media and encoding at least a portion of a synchronization operation, the product comprising program instructions configured to implement:

a snapshot sequence, comprising snapshotting a plurality of application values corresponding to a respective plurality of targeted memory locations to determine whether any of the application values are changed between two successive reads at the targeted memory locations, wherein said snapshotting comprises reading each of the plurality the application values at least twice and comparing each application values across pairs of successive reads;

a linearizable multi-compare, single swap synchronization, comprising updating a first application value corresponding to a first targeted location only if said snapshotting indicates that the plurality of application values remain unchanged between two successive reads at the targeted locations, wherein the first targeted memory location is distinct from the plurality of targeted locations; and

wherein the multi-compare, single swap synchronization further comprises employing a pair of single-location synchronizations to ensure that the application value corresponding to the first targeted location remained unchanged at a linearization point of the snapshot.

wherein the snapshot sequence collects (i) application values associated with each of the other targeted locations and (ii) tagged ids, if any, from corresponding tagged id locations until two successive collections indicate identical respective application values and identical respective tagged ids.

118. (Canceled)